APIs and emerging economy – driving digital transformation through e-government

Denis Trček[1,*]

[1]University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, Ljubljana, Slovenia

Abstract. Digital transformation is increasingly determining the development of societies through ubiquitous deployment of modern information technologies. One of the main drivers that are still not paid sufficient attention are application programming interfaces (APIs). These are not essential just for new services development and adoption, but have further reach and may result even in creation of new industries. Their importance is therefore not to be overlooked for further development, especially by taking into account that the main focus is still on developers (i.e. bottom-up approach). However, higher level business views (i.e. top-down approach) are to be considered in de facto and de iure APIs development, deployment and standardization processes, which is currently not the case. Therefore this paper presents a framework for facilitating APIs (services) evolution by considering top-down business views and their proper addressing. The approach builds on lessons learnt with complex services architectures, and their higher-level derivatives. In line with these lessons it defines implementation strategies at technological and business levels. The whole contribution is conceptualized around e-government services, because governments are key players in each and every economy, and their impact in digital transformation is therefore vital.

1 Introduction

Nowadays, digital transformation is a reality that is driving not only traditional, tangible products focused primary and secondary sectors, but it extends all the way up through the tertiary to the quaternary sector with increasing number of advanced services. And governments present a central entity of digital transformation in quaternary sector. The reasons are rather straightforward.

Governments are typically one of the largest entities in national economies. Their budgets present significant – often major – portion of a country's gross-domestic product (in case of Germany, for example, the current federal budget presents approx. 11% of its yearly GDP [1]). Despite this, when it comes to IT they are too often considered as entities that "passively" support citizens and businesses, while, in fact, they are drivers of whole e-economies:

1. First, governments play considerable roles in all of e-business relations, be it administration to business (A2B) and vice versa (B2A), administration to citizens (A2C) and vice versa (C2A), or administration to administration relations (A2A).
2. Second, considering their potential, they should play a major and active role also in further technology promotion through services they offer, e.g. e-government.

Currently, within the on-going digital transformation, application program interfaces, or APIs, play a pivotal role that exceeds their anticipated influence. Initially and still today, APIs are almost completely considered to be in the domain of developers, i.e. belonging to the technological domain (as stated in [2], "unlike past trends that market to business leaders, APIs market directly to developers"). However, the gathered evidence shows that they do not have significant impact only on the development on new services, but even creation of new industries through new business models [3].

Thus APIs are far more than pure technological artefacts. They affect business even at strategic levels and have to be treated accordingly. Put another way - the gap between top-down, business views focused approaches (e.g. e-government) and bottom-up, technology focused approaches (e.g. APIs) has to be bridged in appropriate way. And this is where the main contribution of this paper comes in. In the second section the relevant technological driving forces are analyzed together with historical perspective to include lessons learnt. This line of reasoning is further refined in the third section, where a new approach to integration of light REST (REpresentational State Transfer) and complex SOAP (Simple Objects Access Protocol) services is presented. The approach builds on existing de facto and de jure standards. In the fourth section the proposed management framework built around e-government initiatives is analyzed and discussed. There are conclusions in the fifth section, followed by acknowledgements and references.

* Corresponding author: denis.trcek@fri.uni-lj.si

2 The evolutionary elements behind digital transformation

The era of e-business began in the mid-nineties of the former century when the Internet started to penetrate business domain [4]. It transformed many industries, starting with the services sector and followed later by tangibles producing sectors. New industries appeared based on new business models (which have transformed in IT inherently present added value), while traditional ones had to adapt many of their processes. As the operationalization and wide implementation of e-business paradigm required new knowledge at the intersection of rapidly evolving IT domain and management domain, a field of e-business engineering emerged [5]. Such approach is crucial, because it enables appropriate addressing of soft and hard factors, which will be also the case in the rest of this paper.

Let’s focus now specifically on services. Already during their early development research focused not only on deployment, but also on their descriptions and discovery. This resulted in specifications of Simple objects access protocol (SOAP), Web services description language (WSDL) and Universal description, discovery and integration protocol (UDDI) [6] (SOAP – WSDL – UDDI triple is also referred to as WS-* family, or remote procedure call, RPC, style services). It is worth to point out that, within this triple, WSDL was actually describing and API for SOAP services.

The SOAP / WSDL / UDDI development was still mainly of technological nature. It was about elementary software procedures (routines) at the business sub-operations level. To further address business needs (i.e. the levels above operations level, all the way to complete business processes), aggregation of these elementary routines was needed for WS-* architectures. But such efforts did not succeed. One well known example was ebXML (e-business eXtended Markup Language) standard proposed by UN CEFAT and OASIS [7], where complex business rules and processes description language were introduced. This was supposed to enable business solutions development with the business processes specifications, which could be almost automatically transformable into program code. However, this technology became very complex and it was too demanding in terms of required efforts and resources for their implementations. On top of this, automatic finding and deploying of available services by using UDDI did not take ground.

What are the lessons learnt for APIs framework? First, avoid too complex structures. Second, avoid imposing too strict specifications – preferably, these should be flexible and based on de iure and de facto solutions to a maximal possible extent. Additional argument for these two requirements is a huge success of lightweight and easier deployable REST services that are forming RESTful architectures. Business community started to deploy them extensively soon after their introduction, and access to these services started almost at the same time to via APIs. As shown in Fig. 1 APIs deployment shows (close to) exponential growth.

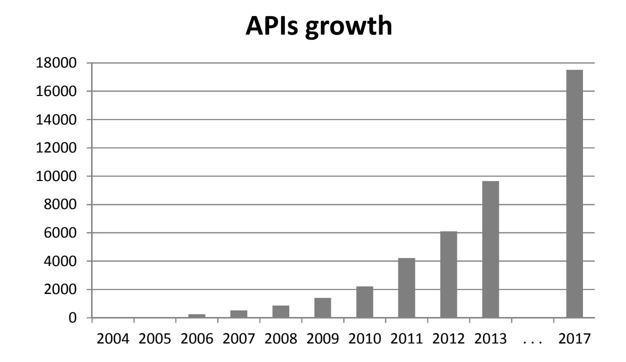


Fig. 1. The growth of REST APIs (source https://www.programmableweb.com).

However, the basic nature of RESTful architectures is such that they and their APIs are developers focused, most likely due to the fact that APIs are protocols intended to be used as interfaces by software components to communicate with each other in order to extend reach of their applications (services) [8]. This narrow view triggered some authors to start exposing business importance of APIs. In [9] it is even exposed that APIs should be treated as a kind of contracts, which are linking the technological and business domains.

3 Fostering business-centric APIs through e-government services

Contrary to SOAP architectures, RESTful APIs have been primarily considered in a data-centric way so far. One understandable reason is due to many governments’ goal that public data should be publicly available. RESTful solutions with their APIs come very handy to fulfill this goal, so it should not be surprising that also OECD in its Open Government Data document, when mentioning APIs, considers them in a data-centric way [8] (this subject is similarly handled in [10]).

Such data-centrism of REST APIs is also a natural consequence of the fact that these web services enable clients to access and manipulate textual representations of resources, i.e. data. Taking into account further that they are largely deployed in inter-organizational settings, while SOAP architectures remain notably limited to intra-organizational settings, there exists a gap. On one hand we have processes centric architectures that are very complex and limited to intra-organizations use, while on the other hand data centric architectures face huge inter-organizational success, but they remain limited to data without extensive offering of processes-centric support.

It can be concluded on the basis of the above given facts and the line of reasoning that REST APIs would benefit from supporting process-centric needs (see Fig. 2). However, imposing such addressing comes with a risk. If this is enforced (e.g. by *de jure* standardization processes), the natural bottom-up driven access may be blocked. Therefore, any *de facto* or *de jure* extensions of APIs should be flexible, potentially optional, and done in a way where existing technology can included with minimal coding or reconfiguration effort.

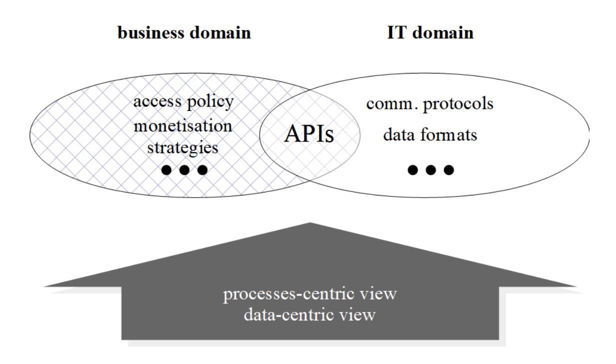


Fig. 2. APIs – linking the technology and business domains.

Before detailing out the proposed extensions to REST APIs, the most important existing standards this area should be briefly provided and analyzed:

- Currently the most promising industry standard, the winning player, is Open API Initiative, OAI, formerly known as Swagger [11]. OAI enables computers to discover and understand a service without accessing its source code or software documentation. This specification that is built upon JSON data representation is comprehensible also to humans, developers and non-developers. OAI is currently the main initiative in the field and it is based on open source software.
- A competing specification to OAI is RESTful API Modeling Language, RAML (http://www.raml.org), which, in turn, is based on a human-readable data serialization language YAML that is a superset of JavaScript Object Notation, JSON. Compared to OAI, RAML specification is supposed to provide more flexibility, even to an extent that includes support of architectures like SOAP [12].
- Beside OAI and RAML, another important specification (that is not an API per se) is Common Gateway Interface, CGI [13]. CGI is the oldest one and during the early ages of web it provided means for running scripts or programs on server's (the most successful language for this purposes was PHP, which is still among the most popular programming languages [14]). Despite popularity of CGI, its standardization never ended in *de iure* standards, although, contrary to a wider belief, CGI still is a de facto standard. It is natively supported by Apache servers, which have close to 50% market share [15]. Therefore, as Apache servers are natively backed by PHP, which may run in CGI mode or as an Apache module, CGI related kind of web services are still more than alive.

Let's now restate our basic problem as follows: Can we make RESTful architecture also processes-aware knowing that this architecture builds upon only PUT, POST, GET, DELETE, HEAD, and PATCH methods, where these methods operate exclusively upon textual resources provided through URIs? If so, the problem now is how to invoke also general procedures and not only data. These are the main options at our disposal:

- Option number one is a new specification (standard) that would use the best from both worlds, RESTful and SOAP, and fuse them.
- Option number two is that SOAP APIs and REST APIs remain as they are and an additional code at a server does the splitting / merging of the services.
- Option number three is to use REST service to accommodate SOAP service, i.e. make SOAP exposed as a REST.
- Option number four is that SOAP is used to accommodate and expose REST service (i.e. becoming incorporated into a SOAP service).
- The fifth option is CGI-BIN based web services deployment.

The first option would enable the best from both worlds, but it is unlikely to happen, as there are currently no such standardization efforts in sight. The second option would also actually pseudo-integrate the two worlds, but would require some adjustments to existing implementations. Nevertheless, it is a doable option, as such adjustments are not excessive (so it is not surprising that certain implementations like Oracle SOA Suite already supports this option in a certain way). The third option is in principle undoable, as RESTful is about textual representations of resources manipulated by REST methods, while SOAP is about any kind of data manipulation and processing procedures (there exists a workaround with limited functionality by wrapping a SOAP call within a REST call). Now as to option four, this option is doable and there do exist such solutions [16], but it conceptually favors SOAP and ineffectively complicates REST parts. Finally, the fifth option is a solution that already exists, although it is tied to two particular technologies, Apache servers and PHP programming language. But these technologies are widely adopted and present *de facto* standard.

According to the above stated analysis, the data and processes focused merging of the two kinds of services can be enabled by using the architectures that are presented in Fig. 3 and Fig. 4. For the first proposed architecture a dispatcher (front-end processor) is introduced. Thus appropriate structuring of a service call needs to be defined. REST services that are nowadays tied to JSON, were initially tied to XML, which is about processes and data. Therefore, as XML technology is no stranger to REST architectures, an efficient way to implement dispatcher architecture goes as follows:

1. Use a minimal http server, where front-end processor is deployed that analyses requests from originators. Its parsing operation relies on appropriate XML schema that reflects composite service envelope structure, presented below.
2. Afterwards parsing, the request is split into REST and SOAP parts accordingly, while each part is forwarded to an intended destination servers.
3. After obtaining responses from the destination, the front-end processor merges the responses in an XML envelope and sends this enveloped content to the originator.

The proposed solution is rather easy to implement with minimal programming, because the majority of tools already exist. For the first step Apache Tomcat server with containers can be used, while another option is implementation of a simple HTTP server that requires

something between fifty to hundred lines of source code. For the second step, a widely available SAX (Simple API for XML) or DOM (Document Object Model) parser can be used. For the third step, XSLT (with XPath) can be used. What needs to be defined for the fourth step is a simple XML wrapper structure, i.e. an envelope for a composite service:

```
<compositeSrvc>
    <RESTservice>…</RESTservice>
    <SOAPservice>…</SOAPservice>
</compositeSrvc>
```

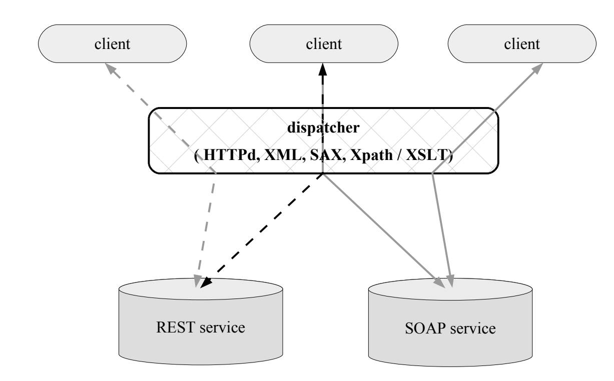


Fig. 3. Dispatcher architecture.

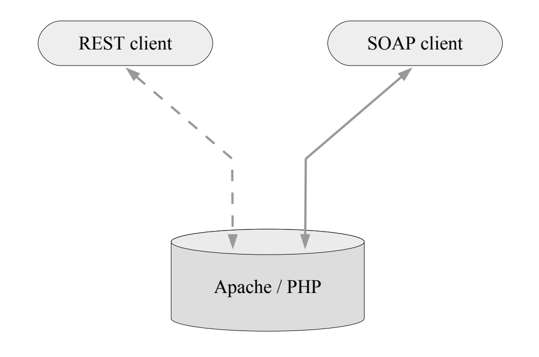


Fig. 4. Apache / PHP architecture.

The second proposed architecture is basically tied to PHP, and the approach is rather straightforward as well. The open source community offers solutions for years, where PHP scripting is integrated with Apache servers. Consequently, provisioning REST and SOAP is just a matter of proper installation and configuration of Apache / PHP pair.

4 Leveraging bottom-up with top-down approaches

Although the main-stream way of thinking about APIs nowadays is data-centric, there are quite some caveats why such view is likely insufficient for a general digital transformation. And this is the point where governments with their e-government services can make a critical technological push.[a] Why such a push is needed, and how to approach it, is further elaborated next.

The first reason is purely technical – the emerging era of the internet of things, IoT, where many devices will lack computing resources, requires more than just raw data. These devices will be forced to "outsource" also significant part of processing. The second reason is misconception that a straightforward utilization of data-centric APIs automatically leads to increased value of these data [19]. The third one is that although APIs are grounded on technological foundations, they have strong organizational implications and influence organizations even at strategic levels [9]. The fourth one is that properly conceptualized APIs enable creation of new business models and even new industries [9, 20], while successful penetration of e-government systems has a notable impact on business value creation, where this penetration depends on technological and (inter)organizational factors [21].

Further, and as already emphasized, for adoption of advanced IT services soft factors are at least as important as hard ones – APIs are no exception.[b] And the core concept that encompasses soft factors is new business model(s). Such models often lead to creation of new industries, not only products and services per se.

Now contrary to common belief that business models are something of importance just to commercial businesses, the truth is that they have lots to do with government agencies and alike as well [22]. Latently present added value in IT technology is released through appropriate business models, be it in commercial sector or public administration. Therefore taking into account the framework presented in this paper we anticipate that e-government provided APIs will also shape business models in general as shown in Fig. 5.

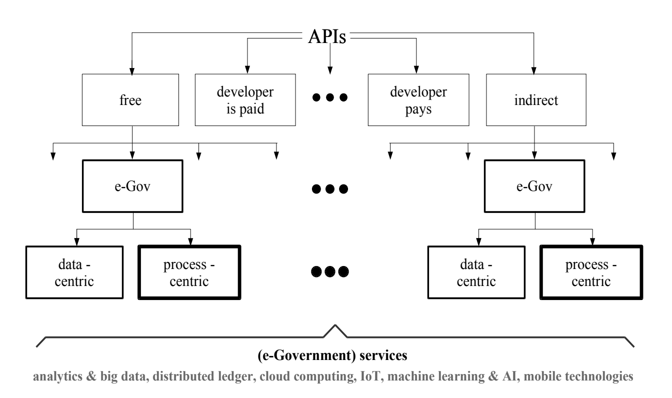


Fig. 5. APIs and e-government driven business models for digital transformation.

5 Conclusions

By following the e-business engineering principles, the approach in this paper focuses on digital transformation. This transformation is typically considered to be driven

[a]One such initiative (and probably the first of this kind) has been implemented very recently by the state of Singapore [17].
[b] Such approach is emphasized also by OECD [18].

by the following technologies [23]: analytics, big data, distributed ledger, cloud computing, internet of things, machine learning with AI, and mobile technologies. Further, it is considered to be driven by the following sectors: banking, consumer products, healthcare, high tech, manufacturing, retail and transportation [23].

However, this paper justifies that digital transformation in tertiary and quaternary sectors is becoming notably tied to APIs (which are front-ends for corresponding services) that are a kind of common denominator of the above technologies. It has been shown that APIs are currently primarily treated in a data-centric way, but they should be considered more broadly to encompass also process-centric views. It is further justified that among the above stated sectors, government sector is certainly among digital transformation drivers and should be included in related efforts – most naturally through e-government services. Such position well coincides with the general position of, e.g. EU Commission about digital scoreboard and related priorities [24, 25].

This presents the basis for the core contribution of this paper, which is architectural framework that relies on de facto and de jure standards to fulfil the above goals: increased digital transformation in services sectors by re-conceptualizing APIs and by focusing on governments' role through their e-government services. More precisely, by building on the influence of governments on many transformation processes, including the digital one, this paper presents a framework that binds business domain with technological domain. It re-conceptualizes the role that APIs currently play and extends it from being primarily about the data to be also about processes. And e-government paradigm with its services is the way to go, where concrete steps are presented that can be implemented with existing solutions and standards – the key accent is focus. Although being a soft factor, appropriate focus of governments has numerous hard consequences. The experience shows that it often enables new business models and creates even new industries. The very basic Internet is one such example and as such additionally justifies the research in the directions given in this paper. However, a large part of research performed so far has often not taken this view into account and has focused on issues like quality of e-government services (see, e.g. [26]).

Speaking purely technologically, the approach presented in this paper is about further services integration and thus presents an evolutionary step forward similar to the steps like front-end and back-end services integration that took place a decade ago [27]. Having integration in mind, future work will address further elaboration of the exposed issues, and eventually some minimal de iure standardization efforts in the area of REST APIs with shifting their focus from dominantly data-centric ones to balanced ones that equally cover processes. Again, if nothing else then the weak processing power segment of the emerging IoT population will stimulate such changes. As a consequence, the importance of security, privacy and safety will play an increased role [28]. Namely, these devices will barely possess required processing power for all possible use and application scenarios, which will therefore have to be harvested from the environment through APIs, even when lightweight solutions are considered.

Author would like to thank to Slovene research agency ARRS, which has supported this research work with founding the research program Pervasive computing, number P2-0359. A part of this research is also the result of collaboration in the ICT COST Action IC1304, Autonomous Control for a Reliable Internet of Services (ACROSS).

References

1. Zimmermann, N.: German federal budget goes up for 2017. Deutsche Welle. http://www.dw.com/en/german-federal-budget-goes-up-for-2017/a-36528845 (2016). Accessed 24 Apr 2019
2. Higginbotham, J.: Designing Great Web APIs – Creating Business Value through Developer Experience. O'Reilly, Sebastopol (2015)
3. Columbus, L.: 2017 is Quickly Becoming the Year of the API Economy. Forbes https://www.forbes.com/sites/louiscolumbus/2017/01/29/2017-is-quickly-becoming-the-year-of-the-api-economy/#1d950ac26a41 (2017). Accessed 30 Mar 2019
4. Trček, D.: Managing Information Systems Security and Privacy. Springer, New York (2006)
5. Kuo-Ming C.: E-services in e-business engineering, Electronic Commerce Research and Applications, **16**(7), 77–81 (2016)
6. Newcomer, E.: Understanding Web Services: XML, WSDL, SOAP, and UDDI, Independent technology guides. Addison-Wesley, Boston (2002)
7. The OASIS ebXML Joint Committee: The Framework for e-Business. OASIS. http://www.oasis-open.org/ (2006). Accessed 24 Apr 2019
8. OECD: Open Government Data. Publications on Digital Government. http://www.oecd.org/gov/digital-government/digital-government-publications.htm (2013). Accessed 21 Mar 2019
9. Jacobson D., Brail, G., Woods D.: APIs: A Strategy Guide. O'Reilly Media, Sebastopol (2011)
10. Tauberer J.: Open Government Data: The Book (2nd Edition). https://opengovdata.io/ (2014). Accessed 24 Apr 2019
11. The Open API Initiative: The Open API Specification v3. The Linux Foundation, https://www.openapis.org/specification/repo (2017). Accessed 24 Apr 2019
12. Sandoval, K.: Top Specification Formats for REST APIs. Nordis APIS. http://nordicapis.com/top-specification-formats-for-rest-apis/ (2015). Accessed 24 Apr 2019

13. Robinson, D., Coar, K.: The Common Gateway Interface (CGI) Version 1.1, RFC 3875. IETF, Reston (2004)
14. Diakopoulos, N.: The 2017 Top Programming Languages: Focus on Jobs. IEEE Spectrum. https://spectrum.ieee.org/computing/software/top-programming-languages-2017-focus-on-jobs (2017). Accessed 24 Apr 2019
15. Netcraftt, February 2017 Web Server Survey, https://news.netcraft.com/archives/2017/02/27/february-2017-web-server-survey.html (2017). Accessed 30 Mar 2019
16. predic8: Tutorial: Exposing SOAP Services as REST Resources. https://www.membrane-soa.org/service-proxy-doc/4.4/rest2soap-gateway.htm (2018). Accessed 30 Mar 2019
17. Basu, M.: Singapore builds government-wide API platform. GovInsider. https://govinsider.asia/digital-gov/singapore-builds-government-wide-api-platform/ (2017). Accessed 30 Mar 2019
18. OECD: Digital Government Strategies for Transforming Public Services in the Welfare Areas, OECD Comparative Study. http://www.oecd.org/gov/digital-government/digital-government-publications.htm (2016). Accessed 30 Mar 2019
19. Tauberer, J.: Open Government Data: The Book. https://opengovdata.io/2014/bulk-data-an-api/ (2014).
20. Caganoff, S.: API Business Models: 20 Models in 20 Minutes. InfoQ. https://www.infoq.com/articles/api-business-models (2013). Accessed 30 Mar 2019
21. Hossain, D., Moon, J., Kim, J.K., Choe, Y.C.: Impacts of organizational assimilation of e-government systems on business value creation: A structuration theory approach. Electronic Commerce Research and Applications. **10(**5), 576–594 (2011)
22. Kaplan, S.: Business Models Aren’t Just for Business. Harvard Business Review. https://hbr.org/2011/04/business-models-arent-just-for (2011). Accessed 30 Mar 2019
23. SAP: Digital Business and Transformation. https://www.sap.com/trends/digital-business.html (2017). Accessed on 30 Mar 2019
24. EU Commission: Digital Scoreboard. https://ec.europa.eu/digital-single-market/en/digital-scoreboard (2017). Accessed 30 Mar 2019
25. EU Commission: Commission and its Priorities, Digital Society. https://ec.europa.eu/commission/tags/digital-society_en (2017). Accessed 30 Mar 2019
26. Soledad, M.J., Miranda, F.J.: Quality in e-Government services: A proposal of dimensions from the perspective of public sector employees. Telematics and Informatics. **35**(2), 457–469 (2018)
27. Glushko, R.J., Tabas, L.: Designing service systems by bridging the “front stage” and “back stage”. Information Systems and e-Business Management. **7**(4), 407–427 (2009)
28. Trček, D., Brodnik, A.: Hard and soft security provisioning for computationally weak pervasive computing systems in e-Health. IEEE Wireless Communications. **20**(4), 22–29 (2013)